Пояснювальна записка
до дипломного проекту
Сипигіна Павла Володимировича
на тему: «Система керування вмістом на базі фреймворку Django»

Київ – 2019 рік

ЗМІСТ

ВСТУП

Аналіз всесвітньої мережі Інтернет, її розвитку та інформаційних систем, що входять у її склад, наразі є дуже актуальним питанням. Особливо через те, що зараз доступ до Інтернету має більше половини населення нашої планети і кількість користувачів росте дуже швидко.

Розвиток технологій передачі даних та мобільного зв'язку призвів до того, що все більше і більше сфер діяльності перетинаються зі сферою інформаційних технологій і вона тим самим стає дуже важливою частиною нашого життя.

Сьогоденні реалії такі, що кожна країна, місто, організація та майже кожна людина має сторінку у Всесвітній мережі. Телекомпанії та паперові видання або дублюються, або ж навіть повністю переходять в інтернет формат. Крім цього, доступність технологій передачі даних і цифрових пристроїв дають змогу широким масам публікуватись в Інтернеті.

Також варто відзначити, що поява і широке розповсюдження Інтернету суттєво вплинуло на сферу бізнесу, надаючи можливість автоматизовано вести документообіг, продавати товари, надавати послуги тощо.

Тривалий час створення сайту для звичайної людини було складним завданням, але зараз майже кожний має змогу створити web-сайт з мінімальними знаннями сучасних технологій. Наразі існують конструктори сайтів і системи керування вмістом, які значно полегшують цю роботу.

Через те, що найпопулярніші системи керування вмістом написані мовою програмування PHP, головною метою даного проекту є створення системи керування вмістом на базі фреймворку Django, а отже мовою програмування Python, для того, щоб надати розробникам та користувачам програмне забезпечення, альтернативне до існуючих рішень.

Особистою метою розробки цього проекту є поглиблення знань у сфері інформаційних технологій та набуття навичок створення програмного забезпечення.

Завданням цієї роботи є створення системи керування вмістом на базі фреймворку Django з системою реєстрації та автентифікації, адміністративною панеллю, рівнями доступу до частин сайту, статичними та динамічно створеними сторінками з навігацією. Ще одним завданням до проекту є дотримання усіх фаз життєвого циклу створення програмного забезпечення: опрацювання вимог, проектування, реалізація, тестування та експлуатація, що повинно значно підвищити його якість.

Для досягнення поставленої мети було використано найбільш універсальні методи дослідження, такі як: спостереження, порівняння, узагальнення, аналіз та синтез.

Практичне значення даного проекту полягає у створенні системи керування вмістом, яку можна використовувати як основу для різноманітних блогів, новинних сайтів, порталів, форумів тощо.

Бакалаврський проект включає в себе наступні розділи: вступ, основні розділи, висновки, список використаних джерел із 20 найменувань та 1 додатку. Графічна частина складається з 4 креслеників формату А3. Загальний обсяг – 67 сторінок.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування доцільності розробки

Системи керування вмістом надають можливість створення сайтів різних ступенів складності при мінімальних витратах часу, а отже і коштів.

Історично склалося, що для реалізації такої системи зазвичай використовують мову програмування PHP, яка має багато недоліків, пов’язаних зі складним налаштуванням серверу і, що найголовніше, має проблеми з безпекою, що у наш час неприпустимо.

На противагу цьому є мова програмування Python і фреймворк для створення web-застосунків Django, що не мають цих недоліків. Крім цього, вони мають декілька значних переваг, як-от: простота освоєння, велика кількість спеціалістів, значна швидкість розробки. Навколо цієї мови і цього фреймворку утворилась величезна спільнота, яка, у разі необхідності, може відповісти на питання або надати вирішення будь-якої проблеми стосовно розробки.

1.1.1 Опис та аналіз предметного середовища

Система керування вмістом — це програмне забезпечення, головною функцією якого є надання можливості організації інформаційних ресурсів в Інтернеті або окремих комп’ютерних мережах.

Зазвичай, такі системи будуються на основі клієнт-серверної архітектури. Клієнт-серверна архітектура складається з розподілених застосунків і передбачає взаємодію та обмін даними між ними.

Вона складається з таких компонентів:

- сервери, які надають дані клієнтським програмам, що її запитують;
- клієнти, які використовують дані, що надають сервери;
- мережа, що надає можливість взаємодії між серверами і клієнтами.

Використання системи контролю вмістом варто розглянути з двох сторін: зі сторони розробника сайту та зі сторони користувача системи.

Головною перевагою для розробника сайту є швидкість, яка полягає у тому, що йому не потрібно створювати сайт з нуля. Система керування вмістом вже має весь необхідний функціонал для роботи сайту. Якщо ж цього функціоналу замало, розробник має можливість підключити модулі, створені ним самим або ж іншими. Модулі, створені іншими розробниками, зазвичай, протестовані часом, тому не мають значних помилок у коді, що в свою чергу пришвидшує розробку.

Переваги для користувача полягають у зручному інтерфейсі системи контролю вмістом, який має не потребує спеціальних навичок для використання, а що найголовніше — користування не передбачає обізнаності у сфері програмування.

1.1.2 Аналіз функціональних особливостей системи

Система керування вмістом зазвичай являє собою web-сайт з безліччю сторінок і складною системою їх організації.

Складається система керування вмістом з декількох основних розділів:

- адміністративна панель;
- статичні сторінки;
- динамічно створені сторінки.

Адміністративна панель є частиною системи, що надає можливість редагувати вміст сайту, а також створювати нових користувачів і надавати їм доступ до певних функцій цієї системи, наприклад, до редагування сторінок. Також, в панелі може бути реалізоване керування модулями.

Варто відзначити, що для багатьох систем керування вмістом існують модулі, які змінюють або розширюють її. Модулі можуть бути суто косметичними, тобто такими, що змінюють зовнішній вигляд сайту, а можуть

додавати функціонал, наприклад, додають можливість приймати платежі за якісь товари або послуги на сайті.

Статичні сторінки являють собою частини такі сторінки, які згенеровані за замовчуванням. Прикладом може слугувати головна сторінка сайту.

Динамічно створені сторінки — це сторінки, створені за допомогою адміністративної панелі, які генеруються на основі заданого шаблону, наприклад, для інформаційного порталу такими є сторінки новин.

До функціональних особливостей системи керування вмістом відноситься такий функціонал, який треба реалізувати:

- система реєстрації та автентифікації;
- адміністративна панель;
- рівні доступу до частин сайту;
- навігаційна панель;
- застосунок для керування динамічно створеними сторінками;
- статична головна сторінка;
- статична сторінка з переліком динамічно створених сторінок.

1.2 Призначення розробки

Розроблений продукт призначений для великої кількості різноманітних сфер і може бути використаний як основа для новинних порталів, блогів, онлайн фотогалерей, інформаційних порталів, сайтів організацій і компаній тощо.

1.3 Цілі та задачі розробки

Цілі розробки цього застосунку можна умовно розділити на дві категорії: для розробників і для користувачів системи.

Головна ціль створення цього застосунку для розробників – це підвищення швидкості розробки сайтів та можливість створювати сайти з системами контролю вмісту використовуючи мову програмування Python.

Головна ціль створення цього застосунку для користувачів системи – спрощення використання і редагування сайтів, покращення їх продуктивності обумовлені дружнім інтерфейсом і автоматизацією наповнення сайту. А також, більш високий ступінь безпеки їх сайту.

1.4 Висновок до розділу

Проаналізувавши використання сучасних технологій і систем, що використовуються у web-індустрії, можна зробити декілька висновків.

По-перше, Інтернет має великий вплив на життя сучасної людини, і цей вплив буде тільки зростати. Всесвітня мережа надає можливість спілкуватись, займатися самоосвітою, роботою, розважатись і отримувати доступ до майже будь-якої інформації в цілому.

По-друге, збільшення користувачів всесвітньої мережі Інтернет призводить до збільшення попиту на створення складних сайтів. Для спрощення розробки і адміністрування і були створені системи керування вмістом. Такі системи знайшли застосування у широкому колі сфер, тому їх розробка і вдосконалення є важливим.

По-третє, інструментарій для створення сайтів не є бездоганним і потребує нових рішень та реалізацій. Мова програмування Python нині є найпопулярнішою, а фреймворк Django є одним із найкращих на сьогоднішній день, тому створення застосунків з їх використанням я вважаю обґрунтованим і доцільним.

Також варто зазначити, що безпека і захист персональних даних нині є одним із головних пріоритетів при створенні програмного забезпечення, тому для розробки варто використовувати найсучасніші інструменти і брати активну участь у їх вдосконаленні.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Існуючі системи керування вмістом

Далі будуть розглянуті декілька популярних систем керування вмістом.

2.1.1 Wordpress

Wordpress – це система керування вмістом на базі мови програмування PHP і системі керування реляційними базами даних MySQL. Головні особливостями цієї системи є архітектура, яка використовує модулі і шаблони. Використовується загалом для створення блогів і форумів. [1]

Головною перевагою Wordpress є велика спільнота користувачів, які з її допомогою створюють сайти та керують ними. За роки існування системи було створено багато шаблонів, які змінюють зовнішній вигляд сайтів, які розробляються. Крім цього у наявності є незліченна кількість модулів, які надають можливість додати до сайту різноманітний функціонал, наприклад, керування багатомовністю сайтів або спостереження за показниками активності користувачів.

Але у даної системи керування вмістом є великі проблеми з безпекою, пов’язані з вразливостями, які створюють модулі користувачів, більшість яких допускають SQL ін’єкції або міжсайтовий скриптинг. [2]

Основні переваги:

- велика кількість модулів;
- велика спільнота;
- велика швидкість виконання;
- адаптована для мобільних пристроїв;
- безкоштовна.

Основні недоліки:

- складна для освоєння;
- проблеми з безпекою.

2.1.2 Drupal

Drupal – це система керування вмістом, призначена для створення блогів, сайтів електронної комерції, соціальних мереж. Як і Wordpress, написана мовою програмування PHP, а для зберігання даних використовує реляційну базу даних MySQL або PostgreSQL. [3]

До головних позитивних особливостей Drupal можна віднести дуже гнучку систему налаштувань, яка дозволяє створювати сайти широкої направленості: від новинних ресурсів до інтернет-магазинів і соціальних мереж. Drupal також має велику спільноту, тому для неї створено безліч готових рішень та інструментарію. У Drupal існують стандарти написання модулів, тому розробники пишуть код для модулів у єдиному стилі, що призводить до того, що розібратись у вже створених модулях доволі легко.

Недоліки Drupal витікають з її переваг: гнучкість налаштувань ускладнює розробку сайтів, вимагаючи від користувачів поглиблене знання програмування і час на освоєння системи.

Основні переваги:

- гнучкість при налаштуваннях;
- велика спільнота;
- безкоштовна;
- велика кількість модулів;
- стандартизація модулів.

Основні недоліки:

- складний для розуміння інтерфейс;
- необхідність просунутих навичок для програміста;
- проблеми з безпекою.

2.1.3 Joomla

Joomla – це система керування вмістом, головна особливість якої полягає у проектуванні і створенні сайту, що відповідає конкретним потребам. Як і дві

попередні системи написана за допомогою PHP і для зберігання даних використовує MySQL або PostgreSQL, а також MS SQL. [4]

Перевагами Joomla можна вважати те, що налаштувати і змінювати сайт можна не маючи знань у програмуванні. Через те, що Joomla вважають другою за популярністю системою керування вмістом (після Wordpress), для неї також існує велика кількість шаблонів і модулів. У даної системи є багато вбудованих функцій, які підвищують швидкість виконання і продуктивність, як от: кешування, GZIP стискання та оптимізація зображень.

До недоліків Joomla можна віднести складність створення необхідних модулів, а також велика кількість шкідливого програмного забезпечення, яке маскують під модулі. Також, у Joomla існують проблеми з сумісністю старих модулів з новими версіями системи.

Основні переваги:

- вимагає мінімальні навички програмування;
- вбудоване кешування;
- велика кількість модулів;
- стандартизація модулів.

Основні недоліки:

- складність створення модулів;
- проблеми сумісності модулів з новими версіями системи;
- проблеми з безпекою.

2.1.4 Wix

Wix – це конструктор сайтів з функціоналом системи керування вмістом. Головна особливість цього застосунку полягає у тому, що він дозволяє без знань HTML змінювати зовнішній вигляд сайту та має магазин застосунків, які розширюють функціонал конструктора. [5]

До головних позитивних боків Wix можна віднести можливість додавання і налаштування будь-яких елементів сторінки і їх властивостей без написання

жодного рядка HTML. Також, у Wix є велика кількість готових шаблонів, які можуть скоротити час на розробку дизайну або каркасу для нього. Ще однією особливістю цієї системи є те, що є можливість безкоштовного розміщення сайту в Інтернеті.

Проте така свобода у розробці призводить до того, що створення гарного зовнішнього вигляду сайту вимагає знань у дизайні та досвіді користування. Хоча створити сайт за допомогою Wix можна безкоштовно, для доступу до повноцінного функціоналу необхідно оформити підписку, що є доволі значним недоліком цієї системи.

Основні переваги:

- інтуїтивний інтерфейс;
- можливість створити інтерфейс сайту самостійно;
- відсутність необхідності встановлення і налаштування сайту на сервері.

Основні недоліки:

- без знань у розробці дизайну важко створити привабливий сайт;
- умовно безкоштовний, необхідність платити за повноцінний функціонал.

2.1.5 Telegraph

Telegraph – це сервіс для анонімної публікації будь-яких текстів. Позиціонується розробниками як видавничий інструмент, що дозволяє створювати публікації та вставляти різноманітні види вбудованого коду. Має лише три поля введення: заголовок, автор, текст публікації. Використовує бібліотеку Quilljs, яка надає можливість легко створювати розробнику текстовий редактор для стрінки для того, щоб користувач міг створювати тексти з розміткою без знань і втручання в HTML. [6]

Основні переваги:

- мінімалістичний і зрозумілий інтерфейс;

- безкоштовний.

Основні недоліки:

- має ознаки, але не може повністю позиціонувати себе системою контролю вмістом.

2.2 Аналіз і порівняння

Розглянуті застосунки є відображенням того, що система контролю вмістом має різноманітні варіації. Проаналізувавши ці застосунки, можна виділити основні моменти, які мають бути присутні у розробленій системі, а саме:

- інтуїтивний інтерфейс;
- велика швидкість виконання;
- простота налаштування і користування;
- високий ступінь безпеки.

2.3 Висновок до розділу

Для того, щоб створити свій програмний продукт, необхідно детально вивчити існуючі рішення і виділити у них сильні та слабкі сторони. Це необхідно для того, щоб на основі цих даних розробити продукт, який перевершує розглянуті аналоги та не має недоліків, притаманних ним.

Також, варто проаналізувати усі наявні дані, відгуки користувачів стосовно існуючих рішень, для того, щоб ще на стадії розробки не робити помилки, які вже були зроблені розробниками тих систем.

3 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Сценарії використання системи

У додатку Д1 наведена діаграма варіантів використання (діаграма прецедентів).

Таблиця 3.1 – Варіанти використання системи

Шифр варіанту використання	Назва варіанту використання
UC-1	Перегляд усіх статей (All posts).
UC-2	Перегляд меню авторизації редактора.
UC-2.1	Авторизація редактора.
UC-2.1.1	Додати статтю (Add post).
UC-2.1.2	Редагувати статтю (Edit post).
UC-2.1.3	Видалити статтю (Delete post).
UC-3.1.4	Вихід з системи редагування (Log out).
UC-3	Перегляд меню авторизації адміністратора.
UC-3.1	Авторизація адміністратора.
UC-2.1.1	Додати редактора (Add user).
UC-2.1.2	Редагувати редактора (Edit user).
UC-2.1.3	Видалити редактора (Delete user).
UC-2.1.4	Перегляд усіх редакторів.
UC-2.1.5	Вихід з системи адміністрування (Log out).

Усі варіанти використання системи представлені у таблиці 3.1.

У таблицях 3.2-3.15 приведені описи кожного з варіантів використання.

Опис варіанту використання з шифром UC-1 зазначено у таблиці 3.2.

Таблиця 3.2 – Варіант використання UC-1

Назва	Перегляд усіх статей (All posts).
Опис	Користувач, редактор та адміністратор мають змогу переглядати список усіх статей.
Учасники	Користувач, редактор, адміністратор.
Передумови	
Постумови	Відображено список усіх статей.
Основний сценарій	1. Система демонструє посилання “All posts”; 2. Користувач, редактор або адміністратор натискає кнопку “All posts”; 3. Система відображає список усіх статей.
Розширення	

Опис варіанту використання з шифром UC-2 зазначено у таблиці 3.3.

Таблиця 3.3 – Варіант використання UC-2

Назва	Перегляд меню авторизації редактора.
Опис	Редактор може скористатися меню авторизації.
Учасники	Редактор.
Передумови	
Постумови	Система відображає меню авторизації редактора.

Продовження таблиці 3.3.

Основний сценарій	1. Система демонструє посилання для переходу на сторінку авторизації; 2. Редактор натискає на посилання для переходу на сторінку авторизації; 3. Система демонструє сторінку авторизації редактора.
Розширення	

Опис варіанту використання з шифром UC-2.1 зазначено у таблиці 3.4.

Таблиця 3.4 – Варіант використання UC-2.1

Назва	Авторизація редактора.
Опис	Редактор проходить авторизацію в застосунку.
Учасники	Редактор.
Передумови	Редактор перейшов на сторінку авторизації.
Постумови	Система авторизує редактора.
Основний сценарій	1. Система демонструє сторінку авторизації редактора; 2. Редактор заповнює поля вводу; 3. Система демонструє кнопку “Log in”; 4. Редактор натискає на кнопку “Log in”; 5. Система авторизує редактора.

Продовження таблиці 3.4.

Розширення	4.1. Система виявляє, що введені дані некоректні; 4.2. Система демонструє сторінку авторизації редактора з повідомленням щодо некоректно введених даних.

Опис варіанту використання з шифром UC-2.1.1 зазначено у таблиці 3.5.

Таблиця 3.5 – Варіант використання UC-2.1.1

Назва	Додати статтю (Add post).
Опис	Редактор має змогу додати статтю.
Учасники	Редактор.
Передумови	Редактор авторизований у системі.
Постумови	Система додає статтю до бази даних.
Основний сценарій	1. Система демонструє сторінку додавання статті та кнопку додавання; 2. Редактор заповнює поля вводу; 3. Редактор натискає кнопку додавання; 4. Система додає статтю до бази даних; 5. Система перенаправляє редактора на сторінку з усіма статтями.
Розширення	

Опис варіанту використання з шифром UC-2.1.2 зазначено у таблиці 3.6.

Таблиця 3.6 – Варіант використання UC-2.1.2

Назва	Редагувати статтю (Edit post).
Опис	Редактор має змогу редагувати статтю.
Учасники	Редактор.
Передумови	Редактор авторизований у системі.
Постумови	Система редагує статтю в базі даних.
Основний сценарій	1. Система демонструє сторінку редагування статті та кнопку збереження; 2. Редактор редагує вміст статті у полях вводу; 3. Редактор натискає на кнопку збереження статті; 4. Система змінює статтю в базі даних; 5. Система перенаправляє редактора на сторінку зі зміненою статтею.
Розширення	

Опис варіанту використання з шифром UC-2.1.3 зазначено у таблиці 3.7.

Таблиця 3.7 – Варіант використання UC-2.1.3

Назва	Видалити статтю (Delete post).
Опис	Редактор має змогу видалити статтю.
Учасники	Редактор.
Передумови	Редактор авторизований у системі.
Постумови	Система видаляє статтю з бази даних.

Продовження таблиці 3.7.

Основний сценарій	1. Система демонструє сторінку редагування статті та кнопку видалення статті; 2. Редактор натискає на кнопку видалення статті. 3. Система видаляє статтю з бази даних; 4. Система перенаправляє редактора на сторінку з усіма статтями.
Розширення	

Опис варіанту використання з шифром UC-2.1.4 зазначено у таблиці 3.8.

Таблиця 3.8 – Варіант використання UC-2.1.4

Назва	Вихід з системи редагування (Log out).
Опис	Редактор має змогу вийти з системи редагування (Log out).
Учасники	Редактор.
Передумови	Редактор авторизований у системі.
Постумови	Редактор виходить з системи редагування та більше не має доступу до меню керування статтями.
Основний сценарій	1. Система демонструє кнопку виходу з системи; 2. Редактор натискає на кнопку виходу; 3. Система видаляє данні про сесію редактора; 4. Система перенаправляє редактора на сторінку з усіма статтями, на якій відсутній доступ до меню редагування статей.
Розширення	

Опис варіанту використання з шифром UC-3 зазначено у таблиці 3.9.

Таблиця 3.9 – Варіант використання UC-3

Назва	Перегляд меню авторизації адміністратора.
Опис	Адміністратор може скористатися меню авторизації.
Учасники	Адміністратор.
Передумови	
Постумови	Система відображає меню авторизації адміністратора.
Основний сценарій	1. Система демонструє посилання для переходу на сторінку авторизації; 2. Адміністратор натискає на посилання для переходу на сторінку авторизації; 3. Система демонструє сторінку авторизації адміністратора.
Розширення	

Опис варіанту використання з шифром UC-3.1 зазначено у таблиці 3.10.

Таблиця 3.10 – Варіант використання UC-3.1

Назва	Авторизація адміністратора.
Опис	Адміністратор проходить авторизацію в застосунку.
Учасники	Адміністратор.
Передумови	Адміністратор перейшов на сторінку авторизації.

Продовження таблиці 3.10.

Постумови	Система авторизує адміністратора.
Основний сценарій	1. Система демонструє сторінку авторизації адміністратора; 2. Адміністратор заповнює поля вводу; 3. Система демонструє кнопку “Log in”; 4. Адміністратор натискає на кнопку “Log in”; 5. Система авторизує адміністратора.
Розширення	4.1. Система виявляє, що введені дані некоректні; 4.2. Система демонструє сторінку авторизації адміністратора з повідомленням щодо некоректно введених даних.

Опис варіанту використання з шифром UC-3.1.1 зазначено у таблиці 3.11.

Таблиця 3.11 – Варіант використання UC-3.1.1

Назва	Додати редактора (Add user).
Опис	Адміністратор має змогу додати редактора.
Учасники	Адміністратор.
Передумови	Адміністратор авторизований у системі.
Постумови	Система додає редактора до бази даних.
Основний сценарій	1. Система демонструє сторінку додавання редактора та кнопку додавання; 2. Адміністратор заповнює поля вводу; 3. Адміністратор натискає кнопку додавання; 4. Система додає редактора до бази даних;

Продовження таблиці 3.11.

Основний сценарій	5. Система перенаправляє адміністратора на сторінку з усіма статтями.
Розширення	

Опис варіанту використання з шифром UC-3.1.2 зазначено у таблиці 3.12.

Таблиця 3.12 – Варіант використання UC-3.1.2

Назва	Редагувати редактора (Edit user).
Опис	Адміністратор має змогу редагувати редактора.
Учасники	Адміністратор.
Передумови	Адміністратор авторизований у системі.
Постумови	Система редагує редактора в базі даних.
Основний сценарій	1. Система демонструє сторінку редагування редактора та кнопку збереження; 2. Адміністратор редагує дані редактора у полях вводу; 3. Адміністратор натискає на кнопку збереження даних редактора; 4. Система змінює дані редактора в базі даних; 5. Система перенаправляє адміністратор на сторінку зі зміненим редактором.
Розширення	

Опис варіанту використання з шифром UC-3.1.3 зазначено у таблиці 3.13.

Таблиця 3.13 – Варіант використання UC-3.1.3

Назва	Видалити редактора (Delete user).
Опис	Редактор має змогу видалити редактора.
Учасники	Адміністратор.
Передумови	Адміністратор авторизований у системі.
Постумови	Система видаляє редактора з бази даних.
Основний сценарій	1. Система демонструє сторінку редактора та кнопку видалення; 2. Адміністратор натискає на кнопку видалення статті; 3. Система видаляє редактора з бази даних; 4. Система перенаправляє адміністратора на сторінку з усіма редакторами.
Розширення	

Опис варіанту використання з шифром UC-3.1.4 зазначено у таблиці 3.14.

Таблиця 3.14 – Варіант використання UC-3.1.4

Назва	Перегляд усіх редакторів.
Опис	Адміністратор має змогу переглядати список усіх редакторів.
Учасники	Адміністратор.
Передумови	Адміністратор авторизований у системі.
Постумови	Відображено список усіх редакторів.

Продовження таблиці 3.14.

Основний сценарій	1. Система демонструє посилання на сторінку з усіма редакторами; 2. Адміністратор натискає на дане посилання; 3. Система відображає список усіх редакторів.
Розширення	

Опис варіанту використання з шифром UC-3.1.5 зазначено у таблиці 3.15.

Таблиця 3.15 – Варіант використання UC-3.1.5

Назва	Вихід з системи адміністрування (Log out).
Опис	Адміністратор має змогу вийти з системи адміністрування (Log out).
Учасники	Адміністратор.
Передумови	Адміністратор авторизований у системі.
Постумови	Адміністратор виходить з системи адміністрування та більше не має доступу до меню керування редакторами.
Основний сценарій	1. Система демонструє кнопку виходу з системи; 2. Адміністратор натискає на кнопку виходу; 3. Система видаляє данні про сесію адміністратора; 4. Система перенаправляє адміністратора на сторінку з усіма статтями, на якій відсутній доступ до меню редагування користувачів.
Розширення	

3.2 Розробка функціональні вимог до системи

Функціональні вимоги до системи є відповідними щодо варіантів використання:

- відображення усіх статей;
- відображення меню авторизації редактора;
- авторизація редактора;
- додавання статті;
- редагування статті;
- видалення статті;
- вихід редактора з системи;
- відображення меню авторизації адміністратора;
- авторизація адміністратора;
- додавання редактора;
- редагування редактора;
- видалення редактора;
- відображення усіх редакторів;
- вихід адміністратора з системи.

3.3 Розробка нефункціональних вимог до системи

Проаналізувавши предметну область та провівши огляд існуючих рішень, можна скласти наступні нефункціональні вимоги до програмного забезпечення:

- розширюваність – система має бути достатньо гнучкою для того, щоб у майбутньому була можливість розширювати функціонал за допомогою створення додаткових модулів;
- переносимість – система має підтримувати перенесення та розгортання її на різноманітних платформах;
- повторне використання компонентів – компоненти системи мають бути розроблені таким чином, щоб у майбутньому була можливість їх повторного використання;

- можливість автоматизування тестування – система повинна мати інструментарій для проведення автоматизованого тестування;
- підтримка сучасними браузерами – увесь функціонал системи має підтримуватись у наступних браузерах: Chrome, Safari, Opera та Firefox;
- безпека – система має використовувати найсучасніші технології забезпечення безпеки та повинна бути стійкою до атак міжсайтового скриптингу та SQL ін’єкцій; також система має зберігати дані редакторів та адміністраторів у захищеному вигляді.

3.4 Висновок до розділу

У даному розділі на базі інформації, що була зібрана та проаналізована у попередніх розділах, було створено діаграму варіантів використань, а також були висунуті функціональні та нефункціональні вимоги до системи. Варто зазначити, що правильно складені вимоги слугують базою для частин майбутньої системи, чим полегшують подальшу розробку та тестування програмного забезпечення.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

4.1 Переваги обраних технологій розробки

При створенні програмних продуктів слід вивчити існуючі технології і практики їх використання для того, щоб обрати для розробки найкращі та найсучасніші з них.

Зазвичай розробка будь-якого застосунку вимагає використання декількох технологій розробки. Наприклад, для створення web-застосунку часто використовують мову розмітки HTML та мову програмування JavaScript для клієнтської частини, а для серверної частини ще якусь мову програмування, як от: PHP, Python, Java тощо. Доволі часто серверна частина може бути написана з використанням декількох технологій, наприклад Facebook реалізована одразу на двох мовах програмування: PHP та C++. [7]

Вибір технології залежить від багатьох факторів пов’язаних як самою системою, так і з командою розробки.

Перший фактор, який впливає на вибір технології – стек навичок команди розробки. У більшості випадків кожен розробник має технології, середовища розробки, мови програмування або розмітки, яким він надає перевагу та з якими він найчастіше працює.

Ще одним фактором при виборі технології є особливості і вимоги до системи, що розробляється. Такими факторами можуть слугувати навантаження на систему, швидкість виконання, сумісність технології з апаратною частиною тощо.

Також варто зазначити, що деякі технології можуть базуватися на інших, використовувати їх або бути сумісними з ними.

При розробці цієї системи керування вмістом були використані наступні технології:

- мова програмування Python;
- фреймворк Django;

- система керування базами даних SQLite;
- мова програмування JavaScript;
- бібліотека jQuery для мови програмування JavaScript;
- бібліотека Quilljs для створення текстових редакторів;
- бібліотека/фреймворк Bootstrap для зміни зовнішнього вигляду web-застосунків;
- веб-сервіс зберігання проектів GitHub;
- веб-сервіс для збору та тестування програмного забезпечення Travis CI.

4.1.1 Python

Python – об’єктно-орієнтована мова програмування високого рівня з динамічною типізацією. [8]

Особливості:

- синтаксис цієї мови дуже простий і короткий, що надає можливість швидко писати код, а також спрощує його читання;
- Python є мовою програмування високого рівня, що означає, що програмісту не потрібно знати системну архітектуру та займатися керуванням пам’яттю;
- мову програмування Python підтримують майже всі сучасні платформи, тому код, написаний на одній з них можна запустити на іншій без змін;
- Python є інтерпретованою мовою програмування, тобто вона виконується рядок за рядком, що може бути корисним для налагоджування коду;
- функціонал цієї мови можна розширити за допомогою підключення бібліотек, написаних іншою мовою (зазвичай C або C++);

- Python має велику кількість вбудованих бібліотек для різноманітних завдань; наприклад, є бібліотеки для роботи з регулярними виразами, базами даних, зображеннями, генерації документації, тестування програмного забезпечення тощо;
- Python – мова з динамічною типізацією, що означає те, що немає необхідно вказувати тип даних заздалегідь.

У даному проекті Python – це основна мова програмування серверної частини.

4.1.2 Django

Django – високорівневий web-фреймворк, написаний на мові програмування Python. [9]

Особливості:

- є дуже добре задокументованим фреймворком, що дозволяє дуже швидко знайти необхідну інформацію про нього;
- написаний на мові програмування Python, яка нині є найпопулярнішою, а також дуже простою у вивченні;
- Django дуже добре масштабується, тому може підійти як для малих проектів, так і для дуже великих; прикладом може слугувати соціальна мережа Instagram, яка генерує величезні обсяги даних кожен день;
- Django дуже безпечний фреймворк, який має багато вбудованих модулів безпеки;
- Django є відкритим програмним забезпеченням, тому у його розробці та вдосконаленні приймають участь багато спеціалістів зі всього світу;
- Django забезпечує дуже швидку розробку через наявність багатьох вбудованих модулів, необхідних для розробки.

Фреймворк Django було обрано для розробки даного проекту через те, що він надає вбудовані базові компоненти для створення системи керування вмістом: модулі для керування базою даних, модулі керування користувачами, модулі для серверної генерації сторінок тощо. Також варто зазначити, що цей фреймворк за замовчуванням використовує сучасні методи для забезпечення безпеки системи, наприклад, токени для запобігання міжсайтових підробок запитів і зберігання хешів замість паролів у відкритому вигляді.

4.1.3 SQLite

SQLite – це система керуванням базами даних на базі мови програмування C. SQLite є вбудованою у всі мобільні пристрої та більшість сучасних комп’ютерів. [10]

Особливості:

- SQLite не потребує ніяких налаштувань;
- уся база даних зберігається в одному файлі;
- підтримує бази даних великого розміру;
- вбудована у більшість сучасних платформ, як от: Android, *BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, WinRT); [11]
- має легкий у використанні інтерфейс програмування застосунків (Application Programming Interface);
- не потребує встановлення додаткових залежностей.

Система керуванням базами даних SQLite була обрана для розробки цього проекту через те, що вона є вбудованою у мову програмування Python, а також через те, що вона зберігає усі дані в одному файлі, що значно спрощує роботу з нею.

4.1.4 JavaScript

JavaScript – це мультипарадигменна мова програмування, яка широко використовується у браузерах для того, щоб надати сторінкам інтерактивності. Використовується також і для написання серверної частини застосунків. [12]

Особливості:

- велика швидкість виконання;
- JavaScript легкий для вивчення та використання;
- велика спільнота розробників та велика кількість готових бібліотек, прикладів коду;
- використання JavaScript може зменшити навантаження на сервер та збільшити швидкість завантаження сторінок;
- додає сторінкам інтерактивності та функціональності;
- у JavaScript постійно виходять оновлення, у яких розширюється його функціонал.

У даному проекті JavaScript є основною мовою програмування клієнтської частини. Також є мовою програмування бібліотек jQuery та Quilljs, використаних при розробці.

4.1.5 jQuery

jQuery – це бібліотека JavaScript, головне застосування якої є зменшення написаного коду і спрощення роботи з елементами web-сторінки. [13]

Особливості:

- jQuery дозволяє проводити маніпуляції з елементами сторінки;
- спрощений синтаксис для обробки подій, наприклад, натискання чи наведення на елемент;
- підтримка AJAX (Asynchronous JavaScript And XML);
- jQuery дозволяє створювати анімації;
- займає дуже мало місця (19 кб при мінімізації і стисненні); [14]
- підтримка старих версій браузерів.

Використання бібліотеки jQuery у цьому проекті скоротило час розробки клієнтської частини і надало можливість зосередитися на інших частинах системи.

4.1.6 Quilljs

Quilljs – бібліотека для створення текстових редакторів для web-сторінок. Перетворює форматований текст з редактора у файл формату JSON для збереження у базі даних і зі збереженого файлу назад в елементи сторінки. [15]

Особливості:

- бібліотека дуже проста у застосуванні та налаштуваннях;
- Quilljs дозволяє використовувати як вбудоване форматування, так і створювати своє;
- підтримка кроссплатформеності: Quilljs генерує однакову розмітку в усіх сучасних браузерах;
- має модулі, які розширюють функціонал, наприклад, додають історію до редактора;
- Quilljs дозволяє налаштовувати зовнішній вигляд редактора відповідно до потреб розробника.

В цьому проекті бібліотека Quilljs була використана як основа для створення текстового редактора системи, а також для перетворення збережених текстів в елементи сторінки.

4.1.7 Bootstrap

Bootstrap – це безкоштовний набір інструментарію для створення сайтів та web-застосунків, який включає в себе готові рішення та шаблони для створення інтерфейсу користувача: блоків сторінки, навігаційних панелей, форм, кнопок, посилань тощо. [16]

Особливості:

- цей інструмент дуже простий у засвоєнні та використанні;

- простий та гнучкий у налаштуванні; хоча Bootstrap і спроектований на створення сітки з 12 колонок, за допомогою вкладення колонок можна досягти бажаного вигляду застосунку;
- Bootstrap був створений для надання сайтам адаптивності, тому не потрібно розробляти дизайн для мобільних пристроїв;
- у наявності є багато шаблонів з використанням Bootstrap, що надає можливість використовувати їх як основу для створення сторінок.

Інструментарій Bootstrap був використаний у даному проекті для пришвидшення розробки клієнтської частини та надання їй уніфікованого та привабливого вигляду.

4.1.8 GitHub

GitHub – це web-сервіс для зберігання проектів, який використовує систему контролю версій Git. Зазвичай використовується командами розробників для сумісною роботою над проектами. [17]

Особливості:

- GitHub має вбудовану систему відстеження проблем та помилок;
- даний сервіс має функцію графічного відображення гілок у проекті;
- GitHub надає можливість для сумісної роботи декількох або навіть великої кількості розробників;
- сервіс має функціонал, що дозволяє розробникам з усього світу пропонувати свої виправлення та покращення коду;
- у сервісі є вбудована можливість надавати права для огляду коду іншими розробникам для забезпечення його якості.

У даному проекті GitHub було використано для збереження коду проекту на серверах сервісу, а також через те, що він сумісний з Travis CI, який буде описано далі.

4.1.9 Travis CI

Travis CI – це сервіс для забезпечення неперервної інтеграції, який використовується для збору та тестування проектів, код яких зберігається на GitHub. [18]

Особливості:

- Travis CI підтримує 21 мову програмування, в тому числі Python та JavaScript, які були використані при розробці даного проекту;
- даний сервіс створює нову віртуальну машину при кожному запуску, що робить тестування більш неупередженим;
- у Travis CI дуже просте налаштування та використання при інтеграції з GitHub;
- також, даний сервіс підтримує кластерне тестування, що пришвидшує виконання тестів і розробку загалом.

Travis CI використовувався при розробці системи для неперервної інтеграції, а саме – для проходження автоматичних тестів при кожному завантажені коду на сервери GitHub.

4.2 Висновок до розділу

У цьому розділі на підставі аналізу предметної області, огляду існуючих рішень, вимог до розроблюваної системи та особистого досвіду було обрано технології, які були використані при розробці даної системи керування вмістом.

5 РОЗРОБКА СИСТЕМИ

5.1 Структура проекту

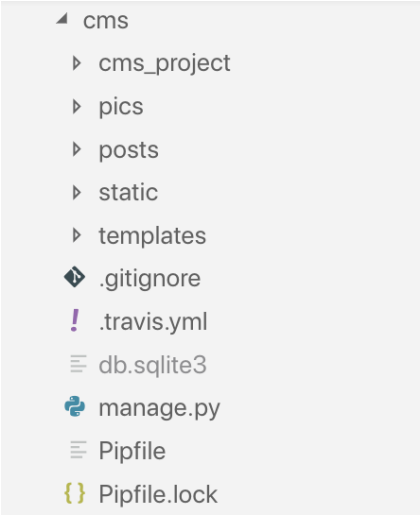

Рисунок 5.1 – структура проекту

Структура проекту, зображена на рисунку 5.1, складається з наступних компонентів:

- cms_project – головний застосунок Django, створений за замовчуванням;
- pics – місце збереження фотографій статей;
- posts – застосунок відповідальний за керування статтями;
- static – набір статичних файлів для клієнтської частини;
- templates – містить шаблони, за якими генеруються HTML-сторінки;
- .gitignore – містить інформацію про файли, які система керування версіями Git має ігнорувати;
- .travis.yml – налаштування для сервісу неперервної інтеграції Travis CI;
- db.sqlite3 – файл, у якому зберігається база даних;
- manage.py – інтерфейс для керування проектами Django, наприклад, для запуску локального серверу або створення нового застонунку;

- Pipfile та Pipfile.lock – файли, необхідні для роботи з пакетом керування віртуальним оточенням Pipenv.

5.1.1 Головний застосунок

Структура головного застосунка зображена на рисунку 5.2:

- settings.py – контролює налаштування усього проекту;
- urls.py – відповідає за те, які види потрібно використовувати для генерації сторінки, яку слід будувати у відповідь на конкретний запит до системи;
- wsgi.py – файл налаштувань інтерфейсу шлюзу веб-сервера (Web Server Gateway Interface).

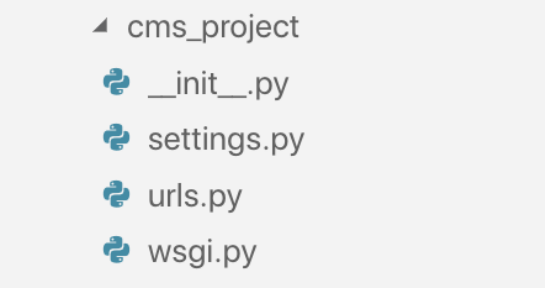

Рисунок 5.2 – структура головного застосунку

5.1.2 Застосунок керування статтями

Стрктура застосунку керування статтями зображена на рисунку 5.3:

- admin.py – відповідає за конфігурацію вбудованого застосунку адміністрування;
- apps.py – містить налаштування самого застосунку;
- forms.py – відповідає за форми, які існують у застосунку;
- models.py – моделі для вбудованої об'єктно-реляційної проекції (Object-relational mapping), на базі яких створюються таблиці в базі даних;
- tests.py – містить автоматизовані тести;

- urls.py – відповідає за те, які види потрібно використовувати для генерації сторінки, яку слід будувати у відповідь на конкретний запит до системи;
- views.py – містить функції видів.

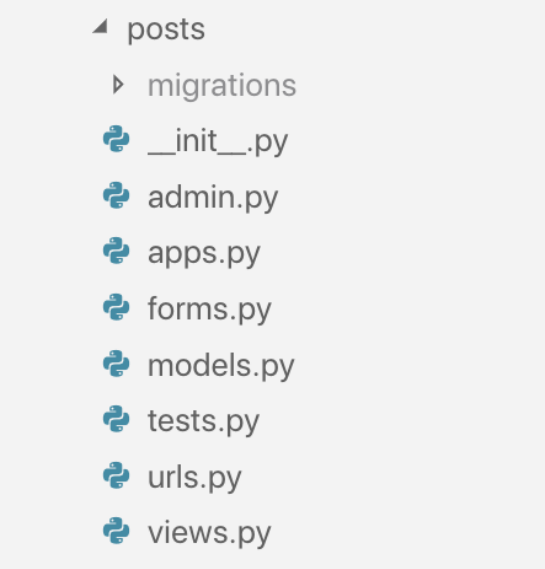

Рисунок 5.3 – структура застосунку керування статтями

Web-застосунок Django працює за наступним сценарієм: застосунок очікує HTTP запит (request) від браузера або іншого клієнта. Коли запит отримано, в залежності від URL та інформації, що міститься у ньому. У залежності від того, що необхідно, застосунок може зчитати або записати інформацію в базу даних, виконати якесь завдання, яке необхідно виконати при отриманні запиту. Після цього застосунок повертає браузеру або клієнту відповідь (response). Зазвичай відповідь містить динамічно згенеровану сторінку з інформацією, яка була отримана раніше при обробці запиту.

Застосунки Django виконують кожний крок у відповідному файлі, як зазначено на рисунку 5.4. Архітектуру Django зазвичай називають MVT («Модель-Вид-Шаблон») подібно до шаблону проектування MVC («Модель-Вид-Контролер»). Варто зазначити, що контролером у Django виступає вид (view), а шаблон (template) виконує функцію виду.

Спочатку HTTP запит направляється до urls.py, мапперу, який перенаправляє його до необхідного виду в залежності від URL, який зазначено

у запиті. Маппер також може звіряти URL з окремими шаблонами символів або цифр, які присутні у ньому та передавати їх як аргументи функції виду.

Далі у файлі views.py HTTP запит направляється у відповідну функцію, яка обробляє його та генерує HTTP відповідь. Зазвичай, вид отримує доступ до необхідної інформації через модель та шаблони.

У файлі models.py описуються об’єкти, на основі яких створюються відповідні таблиці у базі даних, а також механізми, які відповідають за додавання, редагування та видалення з даної бази даних.

Шаблон – це текстовий документ, наприклад HTML, який містить вкраплення спеціального коду, який потім замінюється на інформацію, яка була отримана з моделі, таким чином генеруючи необхідну для HTTP відповіді сторінку.

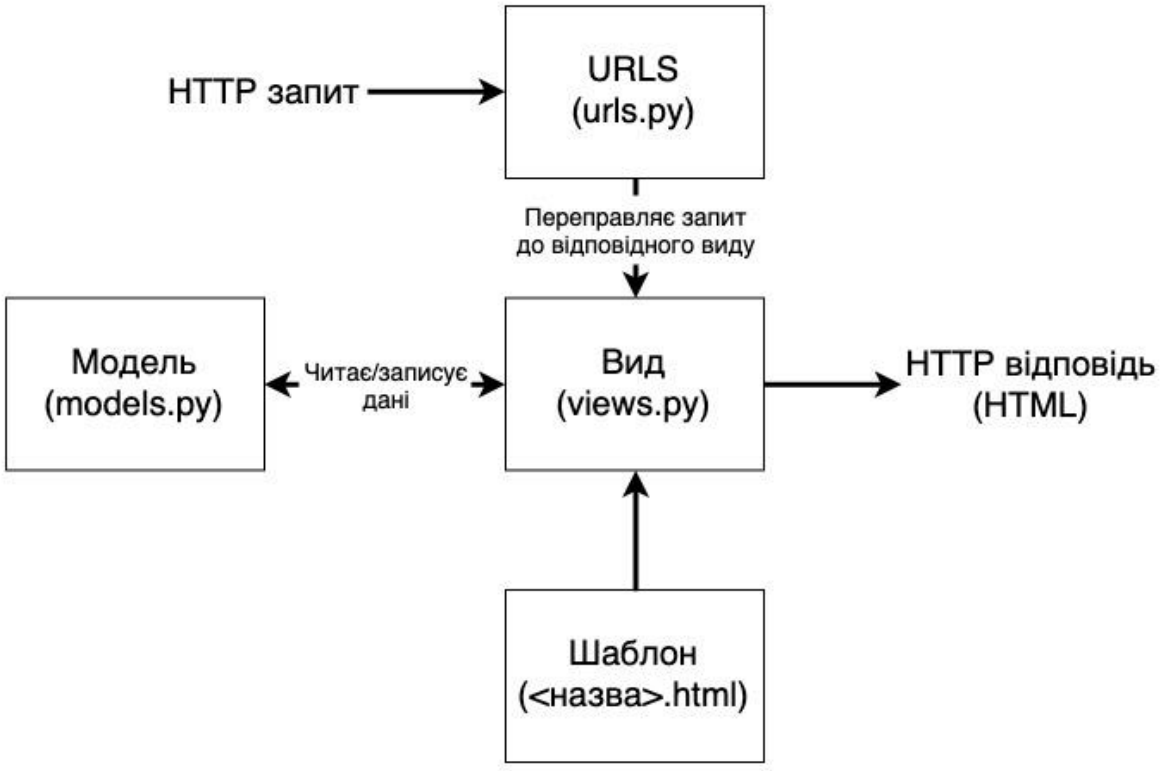

Рисунок 5.4 – Принцип роботи шаблону «Модель-Вид-Шаблон».

У таблиці 5.1 наведено опис моделі застосунку, що відповідає за керування статтями.

Таблиця 5.1 – Опис моделі керування статтями.

Назва моделі	Post(models.Model)
Опис	Модель статті має наступні поля: - pic – відповідає за мініатюру зображення статті, зазначає, що зображення мають зберігатися у папку «pics» та у випадку, коли зображення не обрано, встановлює за замовчуванням зображення з відносним шляхом «pics/None/no-img.jpg»; - title – заголовок статті; має обмеження на довжину в 240 символів; - body – відповідає за збереження основної частини статті; не має обмежень на довжину; - author – зовнішній ключ, що вказує на ідентифікатор користувача, який є автором статті; також, зазначено, що при видаленні користувача, усі статті, які йому належать, будуть видалені каскадно; - date – дата та час публікації; встановлюється автоматично при створенні статті.

Опис видів застосунку наведено у таблицях 5.2-5.7.

Таблиця 5.2 – Опис виду головної сторінки.

Назва методу	posts_home_view(request)
Опис роботи	Вид отримує з бази даних останні 4 додані статті. Потім створює відповідь з головною сторінкою, що їх містить, використовуючи шаблон «home.html».
Використаний шаблон	home.html

Таблиця 5.3 – Опис виду сторінки з усіма статтями.

Назва методу	posts_posts_view(request)
Опис роботи	Вид витягує з бази даних список усіх статей. Після цього генерує відповідь за допомогою шаблону «posts.html», яка містить відображення статей у вигляді блоків, які містять мініатюри зображень та заголовки статей.
Використаний шаблон	posts.html

Таблиця 5.4 – Опис виду сторінки перегляду статті.

Назва методу	posts_post_view(request, id)
Опис роботи	Вид викликає вбудований метод «get_object_or_404», який повертає або об’єкт статті, або одразу генерує відповідь з помилкою з кодом 404 – «сторінка не знайдена». Якщо об’єкт статті знайдено, за допомогою шаблону «post.html» генерується відповідь, яка містить сторінку статті з мініатюрою, заголовком, основною частиною, автором, датою та часом публікації.
Використаний шаблон	post.html

Таблиця 5.5 – Опис виду сторінки редагування статті.

Назва методу	posts_edit_view(request, id)
Опис роботи	Вид викликає вбудований метод «get_object_or_404», який було описано раніше.

Продовження таблиці 5.5.

Опис роботи	Якщо об’єкт статті знайдено та метод запиту «GET», даний метод створює відповідь зі сторінкою редагування статті з формою редагування, що містить наступні поля вводу: зображення мініатюри, заголовок, основна частина та автор статті. Якщо ж метод запиту «POST» – форма дані перевіряються та зміни записуються до бази даних та система генерує відповідь з переадресацією на сторінку з відредагованою статтею. Якщо при перевірці виникли помилки, генерується сторінка редагування з описом помилки, яка виникла. Доступ до цього виду вимагає авторизації у системі.
Використаний шаблон	edit.html

Таблиця 5.6 – Опис виду сторінки видалення статті.

Назва методу	posts_delete_view(request, id)
Опис роботи	Вид викликає вбудований метод «get_object_or_404», який було описано раніше. Якщо об’єкт статті знайдено, запис статті видаляється з бази даних і сервер генерує відповідь з перерадресацією на головну сторінку.

Продовження таблиці 5.6.

Опис роботи	Доступ до цього виду вимагає авторизації у системі.
Використаний шаблон	Відсутній

Таблиця 5.7 – Опис виду сторінки додавання нової статті.

Назва методу	posts_add_view(request)
Опис роботи	Якщо метод запиту – «GET», то вид отримує доступ до необхідної форми, отримує список усіх користувачів з бази даних (для відображення списку можливих авторів статті), та генерує відповідь зі сторінкою додавання статті з формою, що містить наступні поля: зображення мініатюри, заголовок, основна частина та автор статті. Час та дата публікації додається автоматично. Якщо метод запиту – «POST», то отримані з запитом дані спочатку перевіряються, а потім записуються до бази даних. Якщо дані не проходять перевірку, то генерується сторінка додавання статті з описом помилки, яка виникла. Доступ до цього виду вимагає авторизації у системі.
Використаний шаблон	add.html

5.2 Опис роботи системи

У додатку Д2 наведено діаграму компонентів.

З діаграми видно, що розроблена система має клієнт-серверну архітектуру: клієнтом виступає пристрій користувача, а саме – web-браузер, встановлений на ньому. Серверна частина складається з трьох основних компонентів: проксі-сервера Nginx, web-застосунку Django та системи керування базами даних SQLite.

Діаграма послідовності, наведена у додатку Д3, відображає принцип роботи системи, який можна відобразити на прикладі послідовності виконуваних дій при вході користувача у систему:

- спочатку користувач в браузері виконує дію – вводить дані для входу в систему та натискає кнопку входу;
- браузер надсилає запит з даними користувача на необхідну адресу типу «домене_ім’я:80/:443»;
- проксі-сервер перенаправляє даний запит на сервер web-застосунку (Django);
- web-застосунок за допомогою системи керування базами даних звіряє дані з запиту з даними у базі даних;
- після цього web-застосунок генерує відповідь з головною сторінкою або сторінкою входу з повідомленням про помилку, якщо така виникла, та перенаправляє на проксі-сервер Nginx;
- проксі сервер надсилає згенеровану відповідь до браузера;
- браузер відображає отриману сторінку користувачу.

5.3 Розробка бази даних

Особливість фреймворку Django полягає у тому, що для роботи з базами даних він використовує вбудовану об'єктно-реляційну проекцію (Object-relational mapping). Слід відзначити, що Django за замовчуванням створює

таблиці користувачів, груп користувачів, дозволів та допоміжні для них таблиці тощо.

Діаграму структури бази даних наведено у додатку Д4, а опис і призначення таблиць бази даних зазначено у таблиці 5.1.

Таблиця 5.8 – призначення таблиць бази даних.

Назва таблиці	Опис
auth_user	Таблиця зберігає дані про користувача (Редактора, Адміністратора), такі як: ідентифікатор (id), пароль (password), дату останнього входу (last_login), чи є він адміністратором (is_superuser), псевдонім (username), ім’я (first_name), адреса електронної пошти (email), чи є він редактором (is_staff), чи активовано його обліковий запис (is_active), дату його додавання (date_joined), прізвище (last_name).
auth_user_groups	Єднальна таблиця, що зв’язує таблицю користувачів з таблицею груп. Містить у собі власний ідентифікатор (id), ідентифікатори користувача (user_id) та групи (group_id).
auth_group	Таблиця, що відповідає за групи користувачів. Має два поля: ідентифікатор (id) та назву групи (name).
auth_group_permissions	Єднальна таблиця, що з’єднує таблиці груп та дозволів. Містить два три поля: власний ідентифікатор (id), ідентифікатори групи (group_id) та дозволу (permission_id).

Продовження таблиці 5.8.

Назва таблиці	Опис
auth_permissions	Таблиця, що яка містить у собі записи усіх дозволів. Включає в себе чотири поля: ідентифікатор (id), ідентифікатор типу вмісту (content_type_id), кодова назва (codename) та назва дозволу (name).
auth_user_user_permissions	Єднальна таблиця, що відповідає за зв’язок таблиць користувачів та дозволів. Містить три поля: власний ідентифікатор (id), ідентифікатори користувача (user_id) та дозволу (permission_id).
django_content_type	Таблиця, що містить інформацію про тип вмісту. Має три поля: власний ідентифікатор (id), позначка застосунку (app_label) та модель (model).
django_admin_log	Таблиця, яка відповідає за запис та звітування виконаних дій користувачів. Містить у собі наступну інформацію: власний ідентифікатор (id), дату та час виконання дії (action_time), ідентифікатор об’єкту (object_id), представлення об’єкту (object_repr), повідомлення, що описує дію (change_message), ідентифікатори типу вмісту (content_type_id); користувача (user_id), користувача, що виконав дію (user_id) та флаг дії (action_flag), який відповідає за символ, який будо відображатися поряд з дією у списку, наприклад, напроти дії додавання буде показано позначку плюсу, напроти дії видалення буде відображено

Продовження таблиці 5.8.

Назва таблиці	Опис
django_admin_log	позначку мінусу, а напроти дії редагування – позначку олівця.
posts_post	Таблиця, що відповідає за зберігання та керування створених статей. Має 6 полів: власний ідентифікатор статті (id), заголовок статті (title), основний вміст статті (body), дата та час публікації (date), ідентифікатор автора (author_id) та фото мініатюри статті (pic).
django_session	Зберігає інформацію про сесії користувачів. Містить у собі ключ сесії (session_key), дані сесії користувача (session_data), дату та час закінчення терміну дії сесії (expire_date).
django_migrations	Таблиця, що містить інформацію про міграції застосунку – набір змін моделей Django, які за допомогою вбудованої об'єктно-реляційної проекції, створить необхідні команди для зміни таблиць бази даних. Має 4 поля: власний ідентифікатор міграції (id), назва застосунку (app), назва міграції (name), дата та час застосування міграції (applied).

5.4 Висновки до розділу

У даному розділі було описано структуру розробленої системи, а також надано вичерпну інформацію про компоненти системи, їх взаємодію та принцип роботи.

6 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Фази життєвого циклу програмного забезпечення

Нині розробка якісного програмного забезпечення – це доволі структурований процес, який має свої фази життєвого циклу:

- опрацювання вимог;
- проектування;
- реалізація;
- тестування;
- експлуатація.

Під час першої фази необхідно зібрати та проаналізувати інформацію про аналогічні продукти, вивчити предметну область, та на основі отриманих даних розробити вимоги до програмного забезпечення.

Друга фаза, проектування, вимагає від розробника створення архітектури майбутнього програмного забезпечення, створення діаграм прецедентів, потоків даних, станів тощо.

Після цього настає фаза реалізації, під час якої на основі вимог та різноманітних діаграм, зазначених на попередніх етапах, розробляється програмне забезпечення.

Наступною фазою є тестування. Під тестуванням програмного забезпечення мають на увазі процес перевірки реалізованої функціональності до відповідних вимоги до системи, які були складені на перших фазах життєвого циклу.

Остання фаза, експлуатація, відповідає за впровадження програмного забезпечення та його подальший супровід.

6.2 Тестування програмного забезпечення

Варто одразу зазначити, що тестування програмного забезпечення складається не тільки з власне проведення тестів. Це трудомісткий процес забезпечення якості кінцевого застосунку, який включає такі дії, як: глибинний аналіз і планування, розробка тестових сценаріїв, оцінка критеріїв закінчення тестування, написання звітів, рецензування документації та коду тощо.

Існує два типи тестування, які широко використовуються при розробці програмного забезпечення: ручне та автоматизоване. Основним критерієм вибору між ними є вигода для проекту, яка зазвичай залежить від тривалості та проекту складності проекту. Залежність між тривалістю проекту та коефіцієнту рівня прибутковості або збитковості проведення тестів відображено на рисунку 6.1.

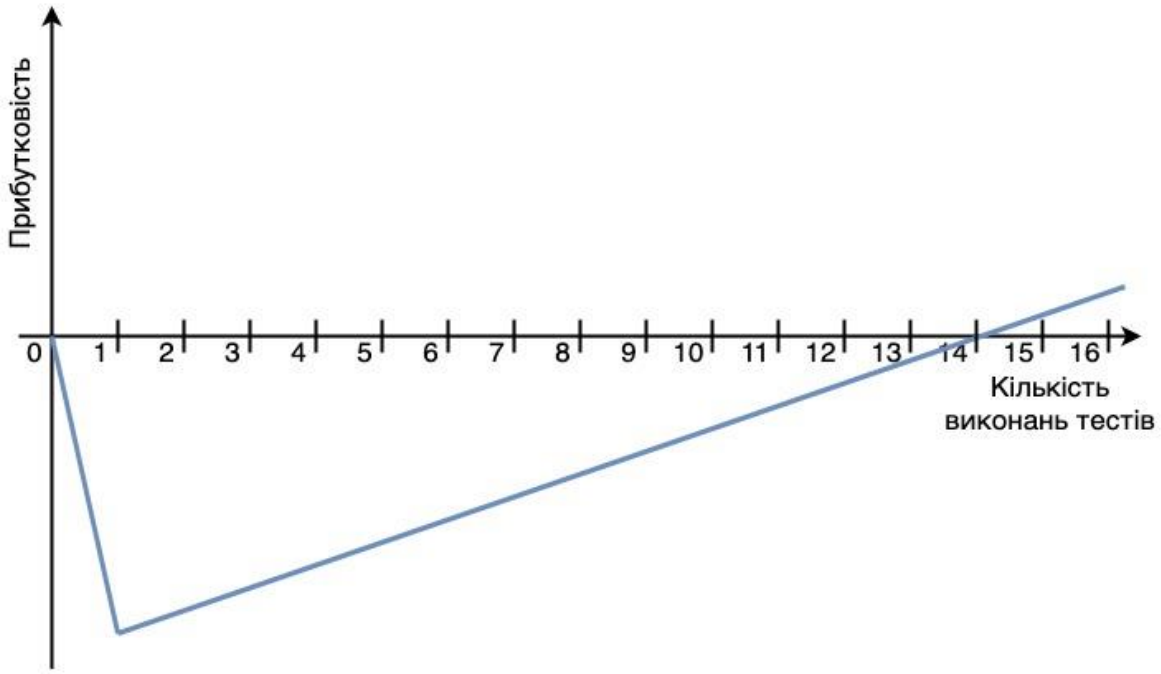

Рисунок 6.1 – Графік залежності між прибутковістю та кількістю виконання тестів

6.2.1 Ручне тестування

При розробці даного програмного забезпечення ручне тестування використовувалося переважно для виявлення помилок у коді. Було використано декілька найпопулярніших методів виявлення помилок. До них належать: метод вставки операторів протоколювання проміжних результатів (logging) для виводу поточних значень змінних у консоль, метод покрокового виконання програми, який надає можливість поетапно переглядати стан програми при її виконанні. Також був використаний метод виконання програми з зупинками (breakpoints), за допомогою якого можна вказати точку програми, при досягнені якої виконання призупиняється для того, щоб можна було розглянути стан програми у зазначений час.

6.2.2 Автоматизоване тестування

Автоматизоване тестування програмного забезпечення полягає у використанні програмних засобів для виконання тестів, перевірки правильності результатів та створення звітів на їх основі.

Для спрощення та структурування розробки автоматизованих тестів складають план тестування в якому описують тестові випадки (test cases) для кожного з окремих тестів.

На основі сценаріїв використання були розроблені тестові випадки та були проведені тести для кожного з них. Тестові випадки та їх результати наведені у таблицях 6.1-6.16.

Таблиця 6.1 – Тестовий випадок «Перевірка відображення сторінки з усіма статтями»

Дія	Відкрити сторінку з усіма статтями.
Очікуваний результат	- сторінка з усіма статтями відкрита; - заголовок сторінки – «Posts – CMS»; - на сторінці відображаються існуючі статті.

Продовження таблиці 6.1

Результат тесту	Пройшов.

Таблиця 6.2 – Тестовий випадок «Перевірка відображення меню авторизації редактора»

Дія	Переглянути меню авторизації редактора.
Очікуваний результат	- сторінка з меню авторизації редактора відкрита; - заголовок сторінки починається з «Log in»; - на формі 2 поля: ім’я та пароль; - кнопка входу доступна.
Результат тесту	Пройшов.

Таблиця 6.3 – Тестовий випадок «Перевірка авторизації редактора з вірними даними»

Дія	Авторизувати редактора з вірними даними.
Очікуваний результат	- кнопка входу натискається; - редактор авторизований після введення вірних даних; - редактор перенаправлений на головну сторінку.
Результат тесту	Пройшов.

Таблиця 6.4 – Тестовий випадок «Перевірка авторизації редактора з невірними даними»

Дія	Авторизувати редактора з невірними даними.
Очікуваний результат	- кнопка входу натискається;

Продовження таблиці 6.4.

Очікуваний результат	- редактор не авторизований після введення невірних даних; - редактор перенаправлений на ту ж саму сторінку; - повідомлення про невірні дані відображається.
Результат тесту	Пройшов.

Таблиця 6.5 – Тестовий випадок «Перевірка додавання статті»

Дія	Додати статтю.
Очікуваний результат	- сторінка додавання статті відкрита; - заголовок сторінки – «Add post - CMS»; - на формі 4 поля: фотографія, заголовок, основна частина та автор; - кнопка додавання статті доступна і працює; - стаття присутня у базі даних.
Результат тесту	Пройшов.

Таблиця 6.6 – Тестовий випадок «Перевірка редагування статті»

Дія	Редагувати статтю.
Очікуваний результат	- сторінка редагування статті відкрита; - заголовок сторінки – «Edit - CMS»; - на формі 4 поля: фотографія, заголовок, основна частина та автор; - кнопка збереження змін статті доступна і працює; - зміни присутні у базі даних.
Результат тесту	Пройшов.

Таблиця 6.7 – Тестовий випадок «Перевірка видалення статті»

Дія	Видалити статтю.
Очікуваний результат	- сторінка редагування статті відкрита; - кнопка видалення статті доступна і працює; - видалена стаття відсутня у базі даних.
Результат тесту	Пройшов.

Таблиця 6.8 – Тестовий випадок «Перевірка виходу з системи редагування»

Дія	Вийти з системи редагування.
Очікуваний результат	- сторінка з меню редагування статей відкрита; - кнопка виходу з системи редагування доступна і працює; - редактор не має доступу до меню редагування статей.
Результат тесту	Пройшов.

Таблиця 6.9 – Тестовий випадок «Перевірка відображення меню авторизації адміністратора»

Дія	Переглянути меню авторизації адміністратора.
Очікуваний результат	- сторінка з меню авторизації редактора відкрита; - заголовок сторінки починається з «Log in»; - на формі 2 поля: ім’я та пароль; - кнопка входу доступна.
Результат тесту	Пройшов.

Таблиця 6.10 – Тестовий випадок «Перевірка авторизації адміністратора з вірними даними»

Дія	Авторизувати адміністратора з вірними даними.
Очікуваний результат	- кнопка входу натискається; - адміністратор авторизований після введення вірних даних; - адміністратор перенаправлений на головну сторінку.
Результат тесту	Пройшов.

Таблиця 6.11 – Тестовий випадок «Перевірка авторизації редактора з невірними даними»

Дія	Авторизувати адміністратора з невірними даними.
Очікуваний результат	- кнопка входу натискається; - адміністратор не авторизований після введення невірних даних; - адміністратор перенаправлений на ту ж саму сторінку; - повідомлення про невірні дані відображається.
Результат тесту	Пройшов.

Таблиця 6.12 – Тестовий випадок «Перевірка додавання редактора»

Дія	Додати редактора.
Очікуваний результат	- сторінка з меню авторизації редактора відкрита; - заголовок сторінки починається з «Add user»;

Продовження таблиці 6.12.

Очікуваний результат	- на формі 3 поля: ім’я, пароль та підтвердження паролю; - кнопка створення доступна і працює; - новий редактор присутній у базі даних.
Результат тесту	Пройшов.

Таблиця 6.13 – Тестовий випадок «Перевірка редагування редактора»

Дія	Редагувати редактора.
Очікуваний результат	- сторінка редагування редактора відкрита; - заголовок сторінки починається з «Change user»; - на формі присутня інформація про редактора; - кнопка збереження змін інформації редактора доступна і працює; - зміни присутні у базі даних.
Результат тесту	Пройшов.

Таблиця 6.14 – Тестовий випадок «Перевірка видалення редактора»

Дія	Видалити редактора.
Очікуваний результат	- сторінка редагування редактора відкрита; - кнопка видалення редактора доступна і працює; - видалений редактор відсутній у базі даних.
Результат тесту	Пройшов.

Таблиця 6.15 – Тестовий випадок «Перевірка відображення сторінки з усіма редакторами»

Дія	Відкрити сторінку з усіма редакторами.
Очікуваний результат	- сторінка з усіма редакторами відкрита; - заголовок сторінки починається з «Select user to change»; - на сторінці відображаються існуючі статті.
Результат тесту	Пройшов.

Таблиця 6.16 – Тестовий випадок «Перевірка виходу з системи адміністрування»

Дія	Вийти з системи адміністрування.
Очікуваний результат	- сторінка з меню адміністрування редакторів відкрита; - кнопка виходу з системи редагування доступна і працює; - адміністратор не має доступу до меню адміністрування редакторів.
Результат тесту	Пройшов.

6.4 Неперервна інтеграція

Неперервною інтеграцією називають практику розробки програмного забезпечення, при якому складання та/або впровадження проекту відбувається автоматизовано. Це необхідно для того, щоб зменшити час та кошти необхідні на дану операцію.

Впровадження неперервної інтеграції полягає у підключені сервісу, який при настанні якоїсь дії, наприклад, запису змін до сховища коду, буде автоматично виконувати задані дії. Зазвичай, це виконання автоматизованого тестування та, при його проходженні, розгортання нової версії програмного забезпечення на вказаному заздалегідь сервері.

При розробці даного програмного забезпечення було використано сервіс Travis CI, який дозволяє синхронізувати проекти, які зберігаються на GitHub та виконувати автоматизоване тестування з мінімальним часом на налаштування.

6.5 Висновок до розділу

Тестування програмного забезпечення – один з найважливіших і трудомістких етапів розробки програмного забезпечення, який доволі часто пропускають заради пришвидшення випуску програмного продукту. Нині існує багато інструментів для полегшення процесу тестування, тому не варто нехтувати цією фазою розробки. Як казав один з розробників Django Джакоб Каплан-Мосс: «Код без тестів – зламаний ще на етапі конструювання». [19]

7 ВПРОВАДЖЕННЯ ТА ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

7.1 Апаратні та програмні вимоги для експлуатації програми

Розроблена система створена за допомогою мови програмування Python, тому апаратні та програмні вимоги до сервера, на якому буде експлуатуватись співпадають з вимогами Python.

Мінімальні системні вимоги до сервера описані у таблиці 7.1, а рекомендовані – у таблиці 7.2.

Таблиця 7.1 – мінімальні системні вимоги до апаратної частини сервера

Процесор	2-ядерний процесор з тактовою частотою 1.3 ГГц або краще.
Місце на диску	2-3 гігабайти.
Оперативна пам’ять	2 гігабайти.
Операційна система	Windows 7 або пізніша, macOS, and Linux.

Таблиця 7.2 – рекомендовані системні вимоги до апаратної частини сервера

Процесор	2-ядерний процесор з тактовою частотою 2.6 ГГц або краще.
Місце на диску	1 гігабайт.
Оперативна пам’ять	4 гігабайти.
Операційна система	Windows 7 або пізніша, macOS, and Linux.

Система створена на базі мови програмування Python версії 3.7, тому для коректної роботи необхідно використовувати цю версію, або ж ту, що була випущена пізніше.

Також, система використовує бібліотеку pipenv для створення віртуального оточення для роботи системи, що забезпечує стабільну роботу системи на будь-якому сервері. Мінімальна рекомендована версія бібліотеки - – 2018.11.26.

Клієнтською частиною системи виступає браузер. Через те, що система використовує JavaScript та його бібліотеки, вимоги до самого застосунку співпадають з вимогами до версій браузерів, починаючи з яких вони підтримуються, що відображено у таблиці 7.3. [20]

Таблиця 7.3 – вимоги до версій браузерів для правильної роботи клієнтської частини

Назва браузера	Мінімальна підтримувана версія
Chrome	73 або вище.
Edge	74 або вище.
Firefox	66 або вище.
Internet Explorer	9 або вище.
Safari (MacOS)	4 або вище.
Opera	60.
Android browser	Версія Android 4.0 або вище.
Safari (iOS)	7 або вище.

Слід зазначити, що клієнтська частина застосунку може працювати ще й у попередніх версіях браузерів, хоча не рекомендується використовувати їх для запобігання неочікуваних результатів і для стабільності роботи системи.

7.2 Інструкція з встановлення

Для встановлення даної системи необхідно відкрити папку зі скопійованим проектом та виконати команду «pipenv install». Ця команда створить віртуальне оточення та встановить усі необхідні бібліотеки для роботи системи. Список залежностей системи зберігається у файлі Pipfile.

Дочекавшись закінчення встановлення бібліотек, слід ввести команду «pipenv shell» для входу у віртуальне оточення, а потім «python manage.py runserver» для запуску серверу.

7.3 Висновки до розділу

Визначення системних та програмних вимог – важлива частина розробки програмного забезпечення. Правильно складена документація щодо цих вимог може зберегти велику кількість часу розробників та користувачів, які будуть працювати з розробленою системою. Також важливим є складання інструкції для встановлення розробленої системи, що надає можливість користуватись програмним забезпеченням не тільки спеціалістам з інформаційних технологій, а й пересічним користувачам комп’ютерів.

ВИСНОВКИ

При виконанні даного дипломного проекту було створено систему курування вмістом на базі фреймворку Django.

Спочатку було проведено аналіз предметної області розроблюваної системи. Було визначено призначення, цілі та задачі розробки, а також було визначено основні особливості майбутньої системи. Далі були проаналізовані існуючі рішення, було створено детальний опис кожного з них. Були розглянуті та проаналізовані переваги та недоліки аналогів. На основі отриманих даних були розроблені функціональні та нефункціональні вимоги до системи. Було проведено вибір технологій розробки, з обґрунтуванням цього вибору. Далі, на основі вимог, визначених раніше, за допомогою обраних технологій було розроблено систему. На наступному етапі було створено тестові випадки та на їх основі проведено тестування. У заключній частині було описано апаратні та програмні вимоги, та вказано інструкцію з встановлення.

Розроблене програмне забезпечення готове до застосування та відповідає усім визначеним вимогам. Також, існує можливість розширення функціоналу застосунку за допомогою модулів, які можна додати потім.

Проаналізувавши існуючі рішення, можна стверджувати, що розроблене програмне забезпечення має низку переваг, до яких можна віднести: більший ступінь безпеки, можливість гнучкого налаштування, використання передових технологій.

Розробка системи керування вмістом саме мовою програмування Python надає можливість збільшити сферу використання даної мови, а також створює альтернативу до існуючих рішень, більшість з яких написана з використанням мови програмування PHP.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brian Messenlehner, Jason Coleman. Building Web Apps with WordPress: WordPress as an Application Framework – O'Reilly Media, 2014 – 462 pages
2. WordPress Vulnerabilities Statistics [Електронний ресурс] – Режим доступу: https://www.wpwhitesecurity.com/statistics-highlight-main-source-wordpress-vulnerabilities/ – Назва з екрану.
3. Todd Tomlinson. Enterprise Drupal 8 Development – Apress, 2017 – 309 pages
4. Dan Rahmel. Advanced Joomla! (Expert's Voice in Web Development) – Apress, 2013 – 412 pages
5. Powerful Features for Your Website | Wix.com [Електронний ресурс] – Режим доступу: https://www.wix.com/features/main – Назва з екрану.
6. Telegraph — Викиреальность [Електронний ресурс] – Режим доступу: www.wikireality.ru/wiki/Telegraph – Назва з екрану.
7. Facebook re-write takes PHP to an enterprise past • The Register [Електронний ресурс] – Режим доступу: https://www.theregister.co.uk/2010/02/02/facebook_hiphop_unveiled/ – Назва з екрану.
8. Mark Lutz. Programming Python: Powerful Object-Oriented Programming – O'Reilly Media, 2011 – 1632 pages
9. Jake Kronika, Aidas Bendoraitis. Django 2 Web Development Cookbook: 100 practical recipes on building scalable Python web apps with Django 2 – Packt Publishing, 2018 – 544 pages
10. Sibsankar Haldar. SQLite Database System Design and Implementation – Sibsankar Haldar, 2015 – 256 pages
11. Features Of SQLite [Електронний ресурс] – Режим доступу: https://www.sqlite.org/features.html – Назва з екрану.

12. Eric Elliott. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries – O'Reilly Media, 2014 – 254 pages
13. Adriaan de Jonge, Phil Dutson. jQuery, jQuery UI, and jQuery Mobile: Recipes and Examples – Addison-Wesley Professional, 2012 – 400 pages
14. jQuery Overview [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/jquery/jquery-overview.htm – Назва з екрану.
15. Why Quill - Quill Rich Text Editor [Електронний ресурс] – Режим доступу: https://quilljs.com/guides/why-quill/ – Назва з екрану.
16. Benjamin Jakobus, Jason Marah. Mastering Bootstrap 4 – Packt Publishing, 2016 – 285 pages
17. Chris Dawson, Ben Straub. Building Tools with GitHub: Customize Your Workflow – O'Reilly Media, 2016 – 302 pages
18. Deepu K Sasidharan, Sendil Kumar N. Full Stack Development with JHipster – Packt Publishing, 2018 – 380 pages
19. A Guide to Testing in Django - Toast Driven [Електронний ресурс] – Режим доступу: toastdriven.com/blog/2011/apr/10/guide-to-testing-in-django/ – Назва з екрану.
20. Browser Support | jQuery [Електронний ресурс] – Режим доступу: https://jquery.com/browser-support/ – Назва з екрану.